class 12

```
library(BiocManager)
library(DESeq2)
```

```
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <-  read.csv("airway_metadata.csv")
```

```
nrow(counts)
```

```
[1] 38694
```

```
metadata
```

```
         id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
2 SRR1039509 treated   N61311 GSM1275863
3 SRR1039512 control  N052611 GSM1275866
4 SRR1039513 treated  N052611 GSM1275867
5 SRR1039516 control  N080611 GSM1275870
6 SRR1039517 treated  N080611 GSM1275871
7 SRR1039520 control  N061011 GSM1275874
8 SRR1039521 treated  N061011 GSM1275875
```

Q1. How many genes are in this dataset? 38694

Q2. How many 'control' cell lines do we have? 4

Q3. How would you make the above code in either approach more robust? Following the below code instead the class website code

```
#control <- metadata[metadata[,"dex"]=="control",]
#control.counts <- counts[ ,control$id]
#control.mean <- rowSums( control.counts )/4
#head(control.mean)
```

```
control.inds <- metadata$dex=="control"
control.ids <- metadata[control.inds,"id"]
control.counts <- counts[,control.ids]
head(control.counts)
```

```
                SRR1039508 SRR1039512 SRR1039516 SRR1039520
ENSG00000000003        723        904       1170        806
ENSG00000000005          0          0          0          0
ENSG00000000419        467        616        582        417
ENSG00000000457        347        364        318        330
ENSG00000000460         96         73        118        102
ENSG00000000938          0          1          2          0
```

```
control.mean <- rowMeans(control.counts)
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.mean)

```
treated.inds <- metadata$dex=="treated"
treated.ids <- metadata[treated.inds,"id"]
treated.counts <- counts[,treated.ids]
head(treated.counts)
```

```
                SRR1039509 SRR1039513 SRR1039517 SRR1039521
ENSG00000000003        486        445       1097        604
ENSG00000000005          0          0          0          0
ENSG00000000419        523        371        781        509
```

```
ENSG00000000457        258        237        447        324
ENSG00000000460         81         66         94         74
ENSG00000000938          0          0          0          0
```

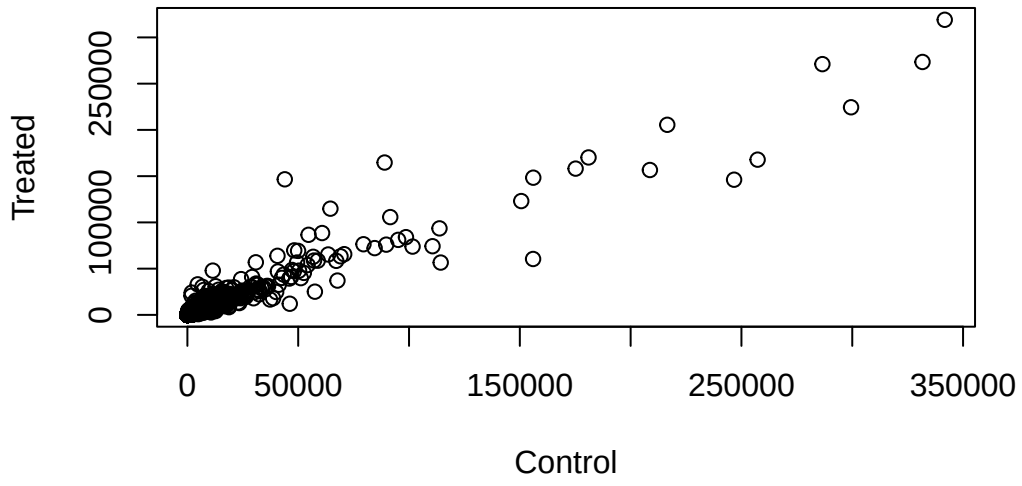
```
treated.mean <- rowMeans(treated.counts)
head(treated.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         658.00            0.00          546.00          316.50           78.75
ENSG00000000938
           0.00
```

```
meancounts <- data.frame(control.mean, treated.mean)
```

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

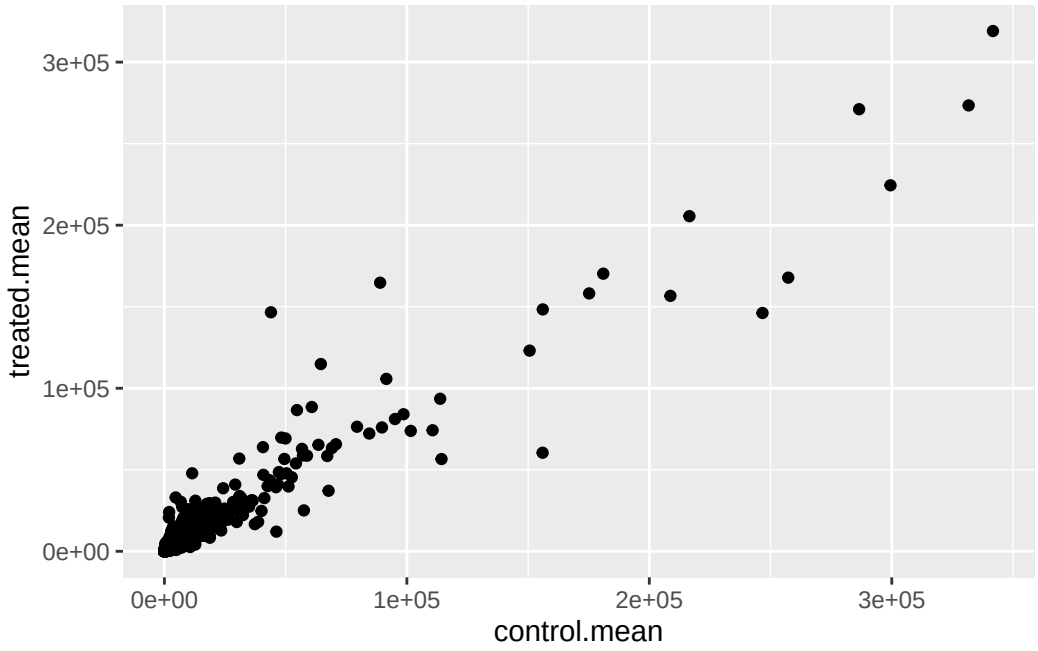
```
plot(meancounts,xlab="Control",ylab="Treated")
```



Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot? geom_point

```
library(ggplot2)
```

```
ggplot(meancounts)+aes(control.mean,treated.mean)+geom_point()
```

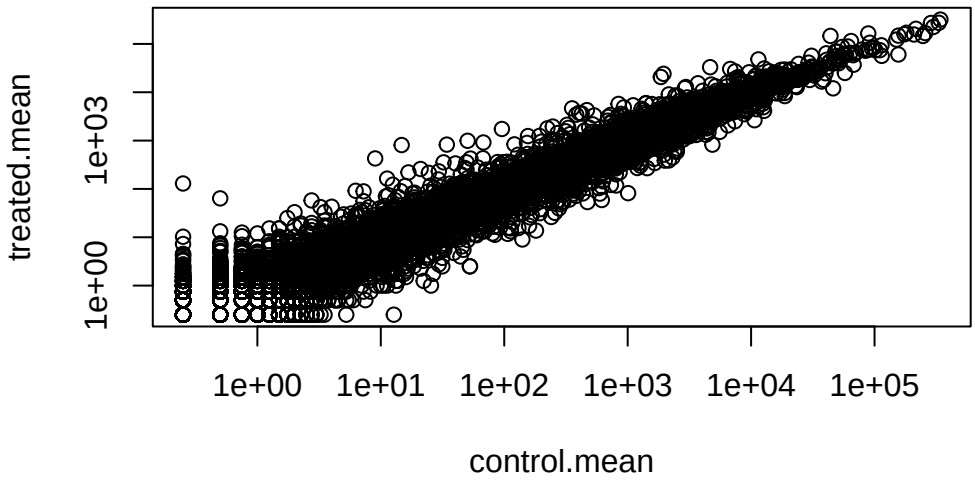


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this? log

```
plot(meancounts,log="xy")
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
from logarithmic plot
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```




```
meancounts$log2fc <- log2(meancounts[,"treated.mean"]/meancounts[,"control.mean"])
head(meancounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000005         0.00         0.00         NaN
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000938         0.75         0.00        -Inf
```

Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function? The arr.ind will cause which() to return all the TRUE values in both the row and column. Calling unique() will ensure we dont count any row twice if it has zer entries in both samples.

```
#zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)

#to.rm <- unique(zero.vals[,1])
#mycounts <- meancounts[-to.rm,]
#head(mycounts)
```

```
to.keep.inds <- rowSums(meancounts[,1:2] == 0) ==0
mycounts <- meancounts[to.keep.inds,]
nrow(mycounts)
```

```
[1] 21817
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level? 250

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level? 367

Q10. Do you trust these results? Why or why not? fold change can be large (e.g. »two-fold up- or down-regulation) without being statistically significant (e.g. based on p-values

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
sum(up.ind)
```

```
[1] 250
```

```
sum(down.ind)
```

```
[1] 367
```

```
sum(mycounts$log2fc >= 2)
```

```
[1] 314
```

```
library(DESeq2)
```

```
dds <- DESeqDataSetFromMatrix(countData=counts,
                              colData=metadata,
                              design=~dex)
```

```
converting counts to integer mode
```

```
Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

```
dds
```

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(1): counts
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
  ENSG00000283123
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(4): id dex celltype geo_id
```

```
dds <- DESeq(dds)
```

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

```
res <- results(dds)
res
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005    0.0000             NA        NA        NA        NA
```

```
ENSG00000000419  520.1342      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457  322.6648      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.6826     -0.1471420  0.257007 -0.572521 0.5669691
...                   ...             ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916      -0.668258   1.69456 -0.394354  0.693319
ENSG00000283123  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1563, 6.2%
LFC < 0 (down)     : 1188, 4.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

```
#plot( res$log2FoldChange,  res$padj)
```

```r
mycols <- rep("grey", nrow(res))
mycols[abs(res$log2FoldChange) > 2]  <- "red"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2)
mycols[inds] <- "blue"


plot( res$log2FoldChange,  -log(res$padj),  col=mycols,
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
abline(v=c(-2,2), col="red")
abline(h=-log(0.05), col="red")
```